

Proprietary Rights Notice: All rights reserved. This material contains the valuable properties and trade secrets of Sirius Systems Technology (SIRIUS) of Scotts Valley California, USA. embodying substantial creative efforts and confidential information, ideas and expressions; no part of which may be reproduced or transmitted in any form or by any means; electronic, mechanical, or otherwise; including photo copying and recording or in connection with any information storage or retrieval system without the permission in writing from SIRIUS.

Copyright notice: Copyright (C) 1981

An unpublished work by Sirius Systems Technology, 380 El Pueblo Road, Scotts Valley, California 95066.

4-SIRIUS COMPUTERS "GRAF" MODULE-

INTRODUCTION:

This document describes the current version of GRAF. GRAF is a graphics utilities package implemented to run under CP/M 1.0/1.3. The utility is designed as an early release version of software given out to programmers who must begin to use the vast graphics capabilities contained within the 800 by 400 resolution screen, the 16 bit micro, and 128k RAM. For demonstration purposes we discuss the use of GRAF from BASIC. The code can, however, be used from other languages. The principles involved will remain similar.

WARNING!

This version of GRAF is designed to be given to users/programmers of the machine to "get you going" with regard to high resolution graphics(hires). The GRAF routines WILL BE CHANGED as we all learn more about the machine. If you wish to incorporate hires into programs that you are developing, look at the way we explain the principles of the GRAF routines(done from a limited BASIC environment). Then, use the language of your choice to write your own graphics routines. Of course, if you wish to use the current GRAF package, you may. It is just that since it will change, we don't want anyone getting a program out that can't grow as the graphics capabilities grow. If your application doesn't require anything more than currently is available you will be fine using GRAF.

SCREEN FORMAT:

The screen is presented as 800 dots(across) by 400 dots(down). Currently two formats are used in addressing the screen. Others may be possible.

-TEXT MODE-

The first mode, called text mode, is the one used by the operating system to place text on the screen. It consists of printing characters on the screen, each character being displayed as a 8x12-dot matrix(8x14 for lower case with decenders). The characters are stored in a 10x16 array in memory. Since the screen is 800 dots wide, this means that you can display 80 characters across the screen. Likewise, 25 characters can go down the display. Since each character on the screen requires 16 bits(2 words) to store, the total memory requirement is 4k(actually 4000 bytes, not 4096). This memory is located in two CMOS chips of 2k each, the first word of each being used together to store the first character, etc.(16 bits each). An individual character consists of 5 attribute bits for inverse video, underline, intensity level, and other features, and 11 bits for the code of the character itself. These 11 bits allow 2048 characters per set, although a normal set might be only 128.

-HIGH RESOLUTION MODE-

This mode is made up of printing (25) 16x16 cells down the screen, with 50 strips going across the screen. The data for the first strip then uses up RAM 2 bytes at a time going down the screen until 25 characters have been stored. The next two bytes in memory represent the TOP character of the next strip to the right, and again the characters are stored in memory going down the strip. The total RAM used is thus: 2 bytes wide per character times 16 bits of height per character times 50 characters wide times 25 characters down; or 40,000 bytes per screen.

HERE ARE THE TWO MODES PICTORALLY:

HOW TO USE:

The necessary files on your demo disk are BBOOT.CMD and GRAF.CMD. BBOOT is the program to run - it accomplishes two things: first, the CP/M top of RAM pointer is moved to allow space for the hires screen and up to 4 character sets; second, the machine code for the graphics routines (GRAF.CMD) is loaded.

Once this is accomplished, BASIC may be loaded. It will return slightly over 10k bytes as free - that is all that is left for the program in this early version of GRAF. Later versions will leave more room for the user's program, but remember that the hires screen by itself uses 40000 bytes of memory. The basic program must set the memory segment to Hex E4E and give the offset for each routine as follows:

```
20 DEF SEG=&HE4E
25 SETTEXT=0
30 SETHIRES=3
35 FILARY=6
40 HDRAW=9
```

```

45 PDUMP=12
50 HPRINT=15
55 CLOAD=18
60 FILL=21
65 CCLR=24
70 CLEN=27
75 HCLR=30

```

The routines are then called from a BASIC program; for example:

```
100 CALL SETHIRES (more specific examples later).
```

In order to directly run a BASIC program that uses hires, form a submit file (say START.SUB) using any editor with the first line: BBOOT, second line: BASIC86 PROGRAM NAME. After booting, simply execute SUBMIT START (the disk cannot be write protected). Your disk should have a file on it called SUBMIT.CMD. When you say "SUBMIT START" CP/M will go out and look for a file called START.SUB and do what it says, line by line. The BASIC86 file used should be the one on the disk. Newer versions may not be compatible with BBOOT when it moves the CPM RAM pointer.

-AVAILABLE COMMANDS-

CALL SETHIRES - enables the hires screen (800 dots horizontally by 400 dots vertically). It is usually a good idea to use the HPRINT call to clear the screen when first enabling hires.

CALL SETTEXT - reverts back to normal character mode (should be followed by a clear screen to get rid of the hires garbage left).

CALL HDRAW(X%,Y%,M%) - Used to draw points and lines on the screen, to clear the screen, and to reverse the screen, ie. to make every green pixel black(off) and every black pixel green. Here are the details of use.

```

X% = x position from 0 to 799
Y% = y position from 0 to 399
M% = mode control.....

```

MODE CONTROL BITS:

76543210

where if bit number

```

7 is on, you will plot chain lines.
6 is on, you will plot ray lines.
5 is on, you will fill the screen.
4 is on, you will move to a new
  position.
(3 & 2 not used)
1 & 0 are for draw control....
  00 = XOR (toggle pixels)

```

01 = OR (green pixels)
10 = AND (black pixels)
11 = XNOR (not used)

MODE CONTROL DESCRIPTION:

The lower 8 bits in the integer M% control the drawing on the screen. Bits 0 & 1 control the way pixels are changed in point, line, and fill screen drawing.

Bit 4 (decimal 16) enables a move to a new position designated by the X% and Y% integers. All other bits are ignored.

Bit 5 (decimal 32) enables the fill screen mode. If M% equals 32+0 all pixels on the screen will be toggled, green to black or black to green. If M% = 32+1 all pixels will be made green. Finally for a value of M% = 32+2, all pixels are made black, ie. THE SCREEN IS ERASED.

Bit 6 (decimal 64) enables ray line drawing. Ray lines are drawn from the last end point to the new X%,Y% position. The last end point remains the same. If M% = 64+0 then XOR lines are drawn (The opposite of whatever pixel color was underneath). If M% = 64+1 then green lines are drawn and for 64+2 black lines will be used.

Bit 7 (decimal 128) enables chain line drawing where the start point for the next line is the end point of the last. This is the same as "DRAWTO" in most BASIC languages. M% = 128+0 gives XOR lines; 128+1 gives green lines; and 128+2 draws with black.

CALL FILARY - This routine is not currently being used and may be ignored in the code.

CALL PDUMP - dumps the contents of the hires screen dot for dot to the Paper Tiger IDS-560 printer. This is done through the standard CENTRONICS port.

CALL HPRINT(X%,Y%,M%) - ORs in the dot pattern for the character specified by M% at the absolute screen location X%,Y%; a 16 by 16 character is assumed with the upper left hand corner of the character placed at the X%,Y% screen location. X coordinates go across the screen and y coordinates go down, with 0,0 at the upper left corner. The X position for the next character is passed back in X% (the routine allows for non-standard width characters - i.e. wider or narrower than 10 dots). The width of the character is specified in the last row of the dot RAM in positions B,C,D and E. If none of those bits are on, a default width of 10 is assumed; otherwise the width is the binary number indicated plus 1 with the LSB in position B, MSB in position E. (For example, a pattern of all four bits on specifies a width of 16; B=1,C=1,D=0,E=1 specifies a width of 12 dots).

HERE IS AN EXAMPLE TO PRINT A STRING(A\$) AT X,Y:

```
10 REM: X,Y ARE COORDINATES OF UPPER LEFT  
20 REM CORNER OF STRING.  
30 REM :CS IS CHARACTER SET NUMBER
```

```

35 A$="THIS IS A SAMPLE STRING."
40 L=LEN(A$):X%=X:Y%=Y
50 FOR HH=1 TO L
60 M%=CS*128+ASC(MID$(A$,HH,1))
70 CALL HPRINT(X%,Y%,M%)
80 NEXT HH

```

CALL CLOAD(S\$,F\$) - Loads a character set from disk file F\$ into memory starting at the location specified by S\$. The locations for full character sets (128 characters) are 0,1,2, & 3, where the S\$ value is calculated as the character set number times 128. (Note: the normal character set is in position 0, if another is loaded into 0, the normal character set must be restored prior to exiting any program). The F\$ string (file name) must have a length of 11 characters (standard CPM file nomenclature, except that the period is not used). All character sets are formed using EDOT.BAS which has been included on the demo disk. The instructions follow the GRAF explanation. Character sets formed with EDOT have been labelled with extents ".CHR" and can be used with CLOAD.

During a program, a given character set can be overwritten (using CLOAD) with another. If this is done in character mode, all screen print will flip to reflect the new character set. This happens because the screen refresh is constantly looking at the shape of the character set as stored in RAM. Change the shapes stored in RAM and the whole screen changes. If done while in HIGHRES, this can be done without affecting any screen portions containing the old character set - i.e. you could display print on the screen in hires with as many character sets as you had. This happens because the HIGHRES screen is stored in a separate area of RAM from where the TEXT mode data is.

A sample call sequence is shown below:

```

100 A$=CHR$(0)
110 FOR I=1 TO 5: A$=A$ + A$: NEXT I
120 DN=0:S%=1*128
130 F$=CHR$(DN)+"FNAME EXT"+A$
140 CALL CLOAD(S$,F$)

```

Lines 100 and 110 set up a null string which is part of the required FCB string. Line 120 sets variable DN to the drive number to load from. A zero value for DN uses the default drive. A one corresponds to drive A, a two to drive B etc. The S\$ value selects the load location, in this example "1". Line 130 puts the FCB string together. "FNAME EXT" may be replaced with any desired file name, but it must be at least 11 characters long, with the extent in the last three characters. If no extent is desired these characters should be spaces. Also note that the "." delimiter is not present. Line 140 makes the actual call. Variable S\$ is returned with a return code of 0 if the load was successful and -1 if the file access was unsuccessful.

Here is another example:

```

10 F$+CHR$(0)

```

```

20 FOR GG=0 TO 5:F$=F$+F$:NEXT
30 L=LEN(FILE$):FOR I = 1 TO 8-L:S$=S$+" ":
  NEXT:FILE$=FILE$+S$
40 REM THE ABOVE LINE FILLS OUT AN INPUTED FILE
  NAME TO 8 LETTERS
50 F$=CHR$(0)+FILE$+"CHR"+LEFT$(F$,24)
60 REM THIS LINE ADDS THE "CHR" EXTENSION TO
  BRING THE TOTAL LENGTH OF THE FILE NAME UP
  TO 11 CHARACTERS & SETS UP F$ FOR USE IN THE ROUT
70 CALL CLOAD(S$,F$)
80 F$=""

```

CALL FILL(X%,Y%,S\$) - fills an enclosed concave area with the pattern specified in S\$ where S\$ is composed of four 16 bit words. For example, if the first word is made up of alternating 1's and 0's and the second of alternating 0's and 1's, the pattern would be every other dot on. The X% and Y% need only specify any point inside the enclosed area. The interior must be entirely off or black. The interior region to be shaded must be a convex set. In other words, it can not curve back on itself. It may be shaped like a circle or square, but not a quarter moon or a region with something inside it. The parameters passed are the X and Y coordinates of any interior point, and a shading pattern string. The shading pattern string is 16 bits wide and 4 bits tall. If all the characters are defined as CHR\$(255) the shade will turn all interior dots on. A shade with a density string of all CHR\$(0) will not do anything. The user must insure that the area is correctly defined with no holes in the boundaries and that the location given is indeed within the region or the routine may malfunction. The following example shades with an alternate dot pattern.

```

100 S$=CHR$(&H55)+CHR$(&H55)+CHR$(&HAA)+CHR$(&HAA)
110 S$=S$+S$:rem double the pattern to fill out S$ to
  the 4 required 16 bit words.
120 CALL FILL(X%,Y%,S$)

```

Here is a blowup of the above pattern:

	x x x x x x x x	H55,H55	= pixel is off
4 BITS	x x x x x x x x	HAA,HAA	x= pixel is on
HIGH	x x x x x x x x	H55,H55	
	x x x x x x x x	HAA,HAA	

16 BITS WIDE

CALL CCLR(X%,Y%,M%) - similar to HPRINT with the exception that the character specified by M% is EXCLUSIVE Ored into the screen. The x and y position of the 16 by 16 character block (upper left hand corner) is passed in the X% and Y% variables, with the M% carrying the location of the character to be exclusive-ored. This routine can be used to erase a single character. For example, assume that the string A\$ contains the letter (in character set 1) to be erased at location 100,32; the BASIC

program should read:

```
100 X%=100:Y%=32:M%=ASC(A$)+1*128:CALL CCLR(X%,Y%,M%)
```

CALL CLEN(M%) - does a lookup of the width of the character specified by M% and returns the width in M%. (Note that the character width must be contained in the character dot ram as described in HPRINT). No screen print is done. This routine can be used in proportionally justifying and/or spacing on the hires screen.

CALL HCLR(X%,Y%,XX%,YY%) - clears the specified rectangular area of the screen with the upper left hand corner specified by X%,Y% and the lower right hand corner specified by XX%,YY%.

COMPLETE EXAMPLES:

The following examples are included on the demo disk that comes with this manual. As with the other materials being given out at this time, these graphics demos are not the "ultimate" examples of programming. Their purpose is to show how to use the early release graphics primitives, ie. to help you to start using high resolution graphics.

EXAMPLE 1 - BARS.BAS

```
10 REM
15 REM      BARS
20 REM      15-OCT-81      LAST LINE #11110
30 rem *****
31 PRINT CHR$(27);"E";:rem clear text screen
32 S1=&HE4E:rem point to start of routines
33 SETTEXT=0:rem give various offsets
34 SETHIRES=3
35 FILARY=6
36 HDRAW=9
37 PDUMP=12
38 HPRINT=15:CLOAD=18:FILL=21:DEF SEG =S1
39 PRINT CHR$(27);"E";
40 rem *****
400 FILE$="BARS":S%=4+0*128:GOSUB 5000:rem FILE$ is the same as
    F$ in the instructions
500 X=12
510 FOR I=0 TO 9
520   X=X+FIX(12*(RND-.35))
530   IF X<0 THEN X=FIX(RND*3)
540   IF X>72 THEN X=72-FIX(RND*3)
550   C1(I)=X
560   X=X+FIX(12*(RND-.35))
570   IF X<0 THEN X=FIX(RND*3)
580   IF X>72 THEN X=72-FIX(RND*3)
590   C2(I)=X
```

```

600 C3(I)=FIX(C1(I)*(.1+.08*(RND-.5)))
610 C4(I)=FIX(C2(I)*(.1+.08*(RND-.5)))
620 NEXT I
700rem *****
1000 REM      bar graph drawer
1010 REM      by Clint Chaplin
1020 REM      9/16/81
1030 B(0)=4+14*128:B(1)=B(0)+1:B(2)=B(0)+2:B(3)=B(0)+3
1050 GOSUB 8000:rem next few lines print text on screen
1070 A$="Bar Graph Demo":X% = 330:Y% = 0:GOSUB 9000
1080 GOSUB 10000
1090 A$="Snoocorp Profit":X% = 320:Y% = 16:GOSUB 9000
1100 REM A$="* Fictitious":X% = 0:Y% = 256:GOSUB 9000
1110 REM A$="  Figures":X% = 0:Y% = 272:GOSUB 9000
1120 A$="Estimated Gross":X% = 250:Y% = 48:GOSUB 9000
1130 A$="Actual Gross":X% = 250:Y% = 64:GOSUB 9000
1140 A$="Estimated Net":X% = 250:Y% = 80:GOSUB 9000
1150 A$="Actual Net":X% = 250:Y% = 96:GOSUB 9000
1155 rem *****
1160 FOR I=0 TO 3
1165 C%=B(I)
1170 FOR J=0 TO 3
1175 X%=450
1180 Y%=48+I*16+J*4
1185 CALL HPRINT(X%,Y%,C%)
1190 NEXT J
1195 NEXT I
2000 FOR I=0 TO 9
2005 B=C1(I)
2010 X% = 200+I*56:Y% = 336
2020 T% = B(0)
2030 GOSUB 11000
2040 NEXT I
2060 FOR I=0 TO 9
2070 B=C2(I)
2080 X% = 228+I*56:Y% = 336
2090 T% = B(1)
2100 GOSUB 11000
2110 NEXT I
2130 FOR I=0 TO 9
2140 B=C3(I)
2150 X% = 200+I*56:Y% = 336
2160 T% = B(2)
2170 GOSUB 11000
2180 NEXT I
2200 FOR I=0 TO 9
2210 B=C4(I)
2220 X% = 228+I*56:Y% = 336
2230 T% = B(3)
2240 GOSUB 11000
2250 NEXT I
3000 REM *****
3010 A$=INKEY$:IF A$="" THEN 3010
3020 M%=34:CALL HDRAW(X%,Y%,M%):rem clear highres screen
3040 CALL SETTEXT:rem back to text mode

```

```

5000 REM    LOAD CHARACTER SUBROUTINE FILE$ IS CHARACTER SET NAME
5001 REM    PASS LOAD LOCATION & FILE NAME
5002 REM    S$ IS CHARACTER SET START ADDRESS
5003 REM    THE CHARACTER SET MUST BE IN DRIVE 2 (OR CHANGE #5015)
5005 F$=CHR$(0)
5010 FOR GG=0 TO 5:F$=F$+F$:NEXT
5012 L=LEN(FILE$):FOR I=1 TO 8-L:S$=S$+" ":NEXT:FILE$=FILE$+S$
5015 F$=CHR$(0)+FILE$+"CHR"+LEFT$(F$,24)
5020 CALL CLOAD(S$,F$)
5025 REM IF S$<>0 THEN PRINT"CHR LD PROBLEM"
5030 RETURN
6000rem *****
8000 REM    highres screen clear
8010 CALL SETHIRES
8020 M%=34:CALL HDRAW(X%,Y%,M%)
8030 RETURN
8500 rem *****
9000 REM    random screen print
9010 REM    A$ is string to print
9020 REM    X%,Y% is the position on the screen to print to
9025 X1%=X%:Y1%=Y%
9030 FOR A1=1 TO LEN(A$)
9040 C% = ASC(A$) + 0*128
9050 A$ = RIGHT$(A$,LEN(A$)-1)
9060 CALL HPRINT(X1%,Y1%,C%)
9070 NEXT A1
9080 RETURN
9999 rem *****
10000 REM    axis draw and label
10010 X%=195:Y%=337:M%=16:CALL HDRAW(X%,Y%,M%)
10020 X%=761:Y%=337:M%=129:CALL HDRAW(X%,Y%,M%)
10030 X%=199:Y%=341:M%=16:CALL HDRAW(X%,Y%,M%)
10040 X%=199:Y%=48:M%=129:CALL HDRAW(X%,Y%,M%)
10050 X%=198:Y%=341:M%=16:CALL HDRAW(X%,Y%,M%)
10060 X%=198:Y%=48:M%=129:CALL HDRAW(X%,Y%,M%)
10070 FOR I=0 TO 10
10080 FOR J=0 TO 3
10090 X%=198+I*56+J:Y%=337:M%=16:CALL HDRAW(X%,Y%,M%)
10100 Y%=341:M%=129:CALL HDRAW(X%,Y%,M%)
10110 NEXT J
10120 NEXT I
10130 FOR I=0 TO 9
10140 FOR J=0 TO 1
10150 X%=195:Y%=48+32*I+J:M%=16:CALL HDRAW(X%,Y%,M%)
10160 X%=199:M%=129:CALL HDRAW(X%,Y%,M%)
10170 NEXT J
10180 NEXT I
10190 FOR I=0 TO 9
10200 X%=208+I*56:Y%=352:A$=RIGHT$(STR$(1981+I),4):GOSUB 9000
10210 NEXT I
10220 FOR I=0 TO 9
10230 X%=170:Y%=330-I*32:A$=RIGHT$(STR$(I*10),2):GOSUB 9000
10240 NEXT I
10250 A$="Year":X%=460:Y%=368:GOSUB 9000
10260 A$="Profit,":X%=40:Y%=192:GOSUB 9000

```

```

10270 A$="Million $":X%=30:Y%=208:GOSUB 9000
10280 RETURN
10999 rem *****
11000 REM      bar drawer
11010 REM      X%,Y% is position of bar
11020 REM      b is height of bar in increments of four lines
11030 REM      t% is type of bar
11040 Y1%=Y%+1
11050 FOR A1=1 TO B
11055   X1%=X%
11060   Y1%=Y1%-4
11070   CALL HPRINT(X1%,Y1%,T%)
11080   X1%=X%+12
11090   CALL HPRINT(X1%,Y1%,T%)
11100 NEXT A1
11110 RETURN

```

EXAMPLE 2 - KEYS.BAS

This example should be tried first using the copy on the demo disk. Keys.bas has many more uses of GRAF than in the first example.

```

25 PI=3.1416
27 AS=1.4:REM      ASPECT RATIO IS AS
30 CFLAG=0:CF=0:CN=0:ERF=0
34 S1=&HE4E:rem set pointer to start of segment
35 SETTEXT=0:SETHIRES=3:FILARY=6:REM next 3 lines define each
routines offset from the start
36 HDRAW=9:PDUMP=12:HPRINT=15
37 CLOAD=18:FILL=21:CCLR=24:CLLEN=27:HCLR=30:DEF SEG = S1
39 PRINT CHR$(27);"E";:rem text mode clear screen
45 CALL SETHIRES:rem setup highres screen
46 GOSUB 700:rem load special character fonts
54 XP=399:YP=190:L=1
55 GOSUB 8000:rem setup softkeys
      Setup strings for printing on highres screen
56 A$(1)="END":A$(2)="PRINT":A$(3)="DRAW CIRCLE":A$(4)="CONTINUOUS":A$
: A$=A$(K)
58 GOSUB 800:rem setup each key
59 NEXT X%=XP:Y%=YP:M%=11+128:CALL HPRINT(X%,Y%,M%):rem print the
11th character of char.set #1 at x%,y%
      Loop to select key and branch
60 K$=INKEY$:IF K$="" THEN 60
61 A=ASC(K$):IF A>48 AND A<58 THEN 2000:rem go to pointer routine
62 IF A=100 THEN 10000:rem go to erase soft key routine & load
char. set
63 IF A>247 OR A<242 THEN 60
64 K=A-240:GOSUB 600:rem reverse video a softkey
66 ON K GOTO 60,1200,1300,1400,1500,1600,1700
169 GOTO 60:rem end of main loop
170 REM      CONTROL CHARACTER ROUTINE
175 A$=INKEY$:IF A$="" THEN 175
180 IF ASC(A$)=13 THEN Y=Y+LF:X=XP:GOTO 1240
185 IF A$="B" AND BFLAG=0 THEN BFLAG=1 ELSE BFLAG=0

```

```

186 IF A$="B" THEN 1240
195 IF A$="D" THEN YP=Y%+4:GOTO 1240
200 IF A$="U" THEN YP=Y%-4:GOTO 1240
215 IF A$="S" THEN CS=1:LF=12:GOTO 1240
217 IF A$="M" THEN CS=2:LF=14:GOTO 1240
219 IF A$="L" THEN CS=3:LF=16:GOTO 1240
234 IF A$="G" THEN X=400-105:Y=0:CS=0:GOSUB 8100:GOTO 1240
235 IF A$="R" AND RFLAG=0 THEN RFLAG=1 ELSE RFLAG=0
237 G=FRE("")
240 GOTO 1240
300 REM CLEAR SCREEN (LEAVE SOFT KEYS ON)
305 REM
310 X%=0:Y%=0:XX%=799:YY%=381:CALL HCLR(X%,Y%,XX%,YY%)
315 X%=XP:Y%=YP:M%=11+128:CALL HPRINT(X%,Y%,M%)
320 GOSUB 600
325 G=FRE(""):GOTO 60
500 REM ERASE SOFT KEY SUBROUTINE
502 X%=XP:Y%=YP:M%=11+128:CALL CCLR(X%,Y%,M%)
505 X%=0:Y%=379:XM%=799:YM%=399:CALL HCLR(X%,Y%,XM%,YM%)
510 RETURN
600 REM REV VIDEO A SOFT KEY
610 REM
620 ST=7+(K-1)*114:EN=109+(K-1)*114
625 M%=10+128:Y%=381
630 FOR X%=ST TO EN
640 X%=X%-1:CALL CCLR(X%,Y%,M%)
650 NEXT
660 RETURN
700 REM LOAD CHARACTER SETS
710 FILE$="SLOGO":CS=0:GOSUB 5000
715 FILE$="SMALL":CS=1:GOSUB 5000
725 FILE$="LGPROPI":CS=3:GOSUB 5000
730 FILE$="MEDPROPI":CS=2:GOSUB 5000
733 G=FRE("")
735 RETURN
800 REM WRITE SOFT KEY LEGENDS
802 REM A$ IS LEGEND K IS KEY NUMBER (1-7)
805 Y=381:XCN=55+(K-1)*114:RM=799:CS=1
810 LE=LEN(A$)
815 X=XCN-3*LE
820 GOSUB 6000
823 G=FRE("")
825 RETURN
900 REM ERASE SOFT KEY - K IS THE NUMBER OF KEY TO ERASE
910 REM
920 X%=6+(K-1)*114:Y%=385:XX%=109+(K-1)*114:YY%=395
930 CALL HCLR(X%,Y%,XX%,YY%)
935 G=FRE("")
940 RETURN
1200 REM PRINT ROUTINE
1205 REM
1210 K$=INKEY$:IF K$="" THEN 1210
1212 IF ASC(K$)=242 THEN 1290
1213 X=XP:Y=YP:X%=XP:Y%=YP
1215 IF K$="[" THEN 170

```

```

1220 A$=K$
1230 GOSUB 6000
1240 A$=INKEY$:IF A$="" THEN 1240
1242 IF ASC(A$)=8 THEN 1280
1245 IF ASC(A$)=242 THEN 1290
1250 IF A$="[" THEN 170
1260 X=X$:SX=X$:Y=YP:GOSUB 6000
1270 P$=A$:GOTO 1240
1280 X$=SX:M$=ASC(P$)+CS*128:CALL CCLR(X$,Y$,M$)
1282 X$=SX
1284 GOTO 1240
1290 RFLAG=0:BFLAG=0:GOSUB 600:GOTO 60
1300 REM CIRCLE ROUTINE
1305 REM
1307 CN=CN+1:ON CN GOTO 1310,1330,1350
1310 GOSUB 900:A$="ENTER CENTER":GOSUB 800:GOSUB 600
1315 GOTO 60
1330 XC=XP:YC=YP
1332 GOSUB 900:A$="ENTER RADIUS":GOSUB 800:GOSUB 600
1350 K$=INKEY$:IF K$="" THEN 1350
1355 N=ASC(K$)-48:IF N>9 OR N<1 THEN 1350
1360 GOSUB 900:A$="R="+STR$(N):GOSUB 800:GOSUB 600
1365 R=N
1370 K$=INKEY$:IF K$="" THEN 1370
1375 N=ASC(K$)-48:IF N=195 THEN GOSUB 7500:GOSUB 900:A$=A$(3):GOSUB 800
1380 IF N>9 OR N<0 THEN 1370
1381 R=10*R+N:IF XC-AS*R<0 OR XC+AS*R>799 OR YC-R<0 OR YC+R>380 THEN 1370
1382 GOSUB 900:A$="R="+STR$(R):GOSUB 800:GOSUB 600
1384 GOTO 1370
1390 GOSUB 900:A$="R TOO LARGE!":GOSUB 800:FOR J=1 TO 500:NEXT J
1392 GOTO 1332
1400 IF CFLAG=0 THEN CFLAG=1 ELSE CFLAG=0
1405 IF CFLAG=1 THEN CF=1
1410 XD=XP:YD=YP
1420 GOTO 60
1500 GOSUB 600:X$=XP:Y$=YP:M$=11+128:CALL CCLR(X$,Y$,M$)
1510 XP=399:YP=190:X$=XP:Y$=YP:M$=11+128:CALL HPRINT(X$,Y$,M$)
1520 GOSUB 900
1530 A$="X="+STR$(0)+" : Y="+STR$(0):GOSUB 800
1540 GOTO 60
1600 A=5:GOTO 2500
1700 REM ERASE RECTANGULAR AREA OF SCREEN
1705 IF ERF=0 THEN ERF=1
1707 ON ERF GOTO 1710,1720,1730
1710 GOSUB 900:A$="UPPER LEFT":GOSUB 800:GOSUB 600
1715 ERF=ERF+1:GOTO 60
1720 UX$=XP:UY$=YP:GOSUB 900:A$="LOWER RIGHT":GOSUB 800:GOSUB 600
1725 ERF=ERF+1:GOTO 60
1730 LX$=XP:LY$=YP:GOSUB 900
1735 CALL HCLR(UX$,UY$,LX$,LY$)
1740 A$=A$(K):GOSUB 800
1742 X$=XP:Y$=YP:M$=11+128:CALL HPRINT(X$,Y$,M$)
1745 ERF=0:GOTO 60
2000 REM POINTER ROUTINE
2010 REM

```

```

2020 A=A-48:IF A=5 THEN 2500
2025 IF CFLAG=0 AND CF=1 THEN CF=0
2030 X%=XP:Y%=YP:M%=11+128:CALL CCLR(X%,Y%,M%)
2040 GOTO 2058
2050 G=PRE("")
2058 ON A GOSUB 2100,2200,2300,2400,2500,2600,2700,2800,2900
2060 A$="X="+STR$(XP-399)+" : Y="+STR$(190-YP):K=5:GOSUB 900:GOSUB
2080 X%=XP:Y%=YP:M%=11+128:CALL CCLR(X%,Y%,M%)
2099 GOTO 60
2100 XP=XP-L:IF XP<0 THEN XP=0
2110 YP=YP+L:IF YP>381 THEN YP=381
2120 GOSUB 2950
2130 RETURN
2200 YP=YP+L:IF YP>381 THEN YP=381
2210 GOSUB 2950
2220 RETURN
2300 XP=XP+L:IF XP>799 THEN XP=799
2310 YP=YP+L:IF YP>381 THEN YP=381
2320 GOSUB 2950
2330 RETURN
2400 XP=XP-L:IF XP<0 THEN XP=0
2410 GOSUB 2950
2420 RETURN
2500 K=6:GOSUB 600
2505 IF L=64 THEN FLAG=0
2507 IF L=1 THEN FLAG=1
2508 IF FLAG=1 THEN L=L*4 ELSE L=L/4
2510 A$="STEP="+STR$(L):K=6:GOSUB 900:GOSUB 800
2520 GOTO 60
2600 XP=XP+L:IF XP>799 THEN XP=799
2610 GOSUB 2950
2620 RETURN
2700 XP=XP-L:IF XP<0 THEN XP=0
2710 YP=YP-L:IF YP<0 THEN YP=0
2720 GOSUB 2950
2730 RETURN
2800 YP=YP-L:IF YP<0 THEN YP=0
2810 GOSUB 2950
2820 RETURN
2900 XP=XP+L:IF XP>799 THEN XP=799
2910 YP=YP-L:IF YP<0 THEN YP=0
2920 GOSUB 2950
2930 RETURN
2950 IF CFLAG=0 THEN RETURN
2960 X=XD:Y=YD:GOSUB 7100:X=XP:Y=YP:GOSUB 7200:XD=X:YD=Y:RETURN
5000 REM LOAD CHARACTER SUBROUTINE FILE$ IS CHARACTER SET NAME
5001 REM PASS LOAD LOCATION & FILE NAME
5002 REM S$ IS CHARACTER SET START ADDRESS
5003 REM THE CHARACTER SET MUST BE IN SELECTED DRIVE (OR CHANGE #5)
5005 F$=CHR$(0)
5010 FOR GG=0 TO 5:F$=F$+F$:NEXT
5012 L=LEN(FILE$):FOR I=1 TO 8-L:S$=S$+" ":NEXT:FILE$=FILE$+S$
5015 F$=CHR$(0)+FILE$+"CHR"+LEFT$(F$,24)
5017 S%=CS*128
5020 CALL CLOAD(S%,F%)

```

```

5025 IF S%<>0 THEN PRINT"CHR LD PROBLEM"
5027 S$=""
5030 RETURN
6000 REM   HIRES PRINT SUBROUTINE A$ IS STRING TO BE PRINTED @ X,Y
6010 LENGTH=LEN(A$):X%=X:Y%=Y
6013 IF RFLAG=1 THEN ST=X%:IF ST=0 THEN ST=1
6015 FOR HH=1 TO LENGTH
6017 TX=X%
6019 M=ASC(MID$(A$,HH,1))
6021 IF M=95 AND X%<RM THEN 6030
6023 IF M=95 AND X%>RM THEN M=45
6024 M%=M+CS*128
6025 CALL HPRINT(X%,Y%,M%)
6027 IF BFLAG=1 THEN X%=TX+1:CALL HPRINT(X%,Y%,M%)
6029 IF X%>RM AND M=32 OR X%>RM AND M=45 THEN X%=LM:Y%=Y%+1:IF HH<>LE
6030 NEXT
6032 IF RFLAG=1 THEN EN=X%:M%=9+CS*128:FOR X%=ST TO EN:X%=X%-1:CALL CE
6035 RETURN
6050 IF ASC(MID$(A$,HH+1,1))=32 THEN HH=HH+1
6051 GOTO 6030
6100 REM
6105 REM           HIRES CHARACTER ERASE SUBROUTINE
6110 REM
6115 LENGTH=LEN(A$):X%=X:Y%=Y
6120 FOR HH=1 TO LENGTH
6125 M%=ASC(MID$(A$,HH,1))+CS*128
6130 CALL CCLR(X%,Y%,M%)
6135 NEXT
6145 RETURN
7000 REM   HIRES SREEN CLEAR SUBROUTINE
7005 M%=34:CALL HDRAW(X%,Y%,M%)
7010 RETURN
7100 REM   HIRES POSITION (NO DRAW) @ X,Y
7105 X%=X:Y%=Y:M%=16:CALL HDRAW(X%,Y%,M%)
7110 RETURN
7200 REM   HIRES DRAW FROM PREVIOUS X,Y TO CURRENT X,Y
7215 X%=X:Y%=Y:M%=129:CALL HDRAW(X%,Y%,M%)
7220 RETURN
7500 REM DRAW CIRCLE
7505 REM
7508 TH=PI/30
7510 IF XC+AS*R>799 OR XC-AS*R<0 THEN RETURN
7515 IF YC+R>380 OR YC-R<0 THEN RETURN
7520 X=XC+AS*R:Y=YC:GOSUB 7100
7525 FOR TT=0 TO 2*PI+TH STEP TH
7530     X=XC+AS*R*COS(TT):Y=YC+R*SIN(TT)
7535     GOSUB 7200
7540 NEXT
7550 RETURN
8000 REM   ENTER SOFT KEY OUTLINE
8005 X%=2:Y%=383
8010 FOR II=1 TO 7
8020     M%=1*128:CALL HPRINT(X%,Y%,M%)
8030     FOR JJ=1 TO 5
8040         M%=1+1*128:CALL HPRINT(X%,Y%,M%)

```

```

8050     NEXT JJ
8060     M%=2+1*128:CALL HPRINT(X%,Y%,M%)
8065     X%=X%+2
8070 NEXT II
8075 G=PRE("")
8080 RETURN
8100 REM
8102 REM     LOGO SUBROUTINE
8104 REM
8106 X%=XP:Y%=YP
8108 FOR I%=0 TO 20
8110     M%=I%+CS*128
8112     CALL HPRINT(X%,Y%,M%)
8114 NEXT
8116 X%=XP:Y%=YP+16
8118 FOR I%=21 TO 41
8120     M%=I%+CS*128
8122     CALL HPRINT(X%,Y%,M%)
8124 NEXT
8126 X%=XP:Y%=YP+32
8128 FOR I%=42 TO 63
8130     M%=I%+CS*128
8132     CALL HPRINT(X%,Y%,M%)
8134 NEXT
8135 X=X%:Y=Y%
8136 RETURN
10000 GOSUB 500:FILE$="NORMAL":CS=0:GOSUB 5000
10005 CALL SETTEXT:PRINT CHR$(27);"E";
10010 END

```

INSTRUCTIONS FOR USING EDOT.CMD

Since EDOT is a command file, it is run by simply going to the system level and saying "EDOT". The program allows you to ~~edit character sets from disk. The disk format is four characters~~ to a sector, binary image. Since only characters are stored in the file, and no partial sectors are used, characters reads and writes must be in multiples of four characters.

The editor has two basic input states. The first encountered is the character number input state. Here you have a choice of the following options (upper or lower case may be used).

- R - READ A FILE: Prompts for a file name and the character location. If just a carriage return is entered for a file name, then the last used file name or the file name used at program invocation is used.
- W - WRITE A FILE: Prompts for file name, size, and starting location.
- ^C - "CONTROL C": Exits to CPM.

{NUMBER} : Entering a decimal number displays that character and enters the edit state.

-EDITING OPTIONS-

- A - Advance to next character number.
- B - Go back one character.
- C - Clear character dots.
- M- Move current character to a second character location. User is prompted for the new location.
- N - Return to character number input state.
- R - RESET MODE - Reset to 0 any dot encountered through cursor movement.
- S - SET MODE - Opposite of reset mode.
- P - POSITION MODE - Allows movement of cursor without effecting dots. Position mode is the default mode for the edit state.
- E - Same as P.
- T- TOGGLE MODE - Flips any location encountered with cursor movement.
- 1 - Move cursor down and left
- 2 - Move cursor down.
- 3 - Move cursor down and right.
- 4 - Move cursor left.
- 5 - Toggle cursor location (does not change mode).
- 6 - Move cursor right.
- 7 - Move cursor up and left.
- 8 - Move cursor up.
- 9 - Move cursor right.

NOTES:

12/18/81 - Starting number must be divisable by 4 (4 characters per sector).

12/18/81 - To save, N then W .